Remarks

Claims 1-12 and 15-16 remain pending. Claim 13-14 and 17-20 are hereby canceled without prejudice to facilitate issuance of the remaining claims. Claims 1 and 15 are hereby amended. No new matter is being added.

Claim Rejections--Section 101

Claims 13-15 were rejected under 35 U.S.C. 101 as not reciting a practical application, where the practical application "can be provided by a physical transformation or a useful, concrete and tangible result." Claim 13 is hereby amended, and claims 14-15 depend from claim 13. This rejection is traversed with respect to the claims as now amended.

As amended, claim 13 now recites as follows. "An operating system with capability to obtain status information from user threads of a target process so as to debug the target process without requiring stopping the target process, the operating system comprising ...." (Insertion shown by underline.) Hence, **claim 13 now recites the result of debugging the target process without requiring stopping the target process**.

Debugging determines coding errors and is very important in software program development. Without debugging, most computer programs would fail to function properly. Therefore, applicant respectfully submits that debugging of a target process is without a doubt a useful, concrete and tangible result. The importance of debugging is evidenced by the fact that there are entire books devoted to the subject, including the book "Windows 2000 Kernel Debugging" by Steven McDowell, which was cited in the office action.

For at least the above-discussed reasons, applicant respectfully submits that amended claim 13 overcomes the rejection under Section 101. Claims 14-15 depend from claim 13. Hence, applicant respectfully submits that claims 14-15 now also overcome the rejection under Section 101.

Claim Rejections--Section 102

Claims 1-15 were rejected under 35 U.S.C. 102 as being anticipated by the McDowell reference ("Windows 2000 Kernel Debugging," 2001). This rejection is respectfully traversed.

The Examiner cites to the McDowell reference and, in particular, to portions of Chapter 5 (Kernel Debugging) and Chapter 6 (Debugging the Hardware) of McDowell. However, as discussed below, the cited portions of McDowell do not read upon the limitations of independent claim 1, nor do they read upon the upon the limitations of independent claim 13.

Claims 1-12

Claim 1 recites as follows.

> 1. A method of obtaining status information from user threads of a target process, the method comprising:
> performing a system call from a querying process;
> **creating a kernel debug thread in a kernel entity of the target process**; and
> **creating a user status thread in a user entity of the target process.**

(Emphasis added.)

As seen above, claim 1 recites, among other limitations, "creating a kernel debug thread in a kernel entity of the target process." Creating the kernel debug thread (KDT) is illustrated, for example, by the arrow labeled "Create KDT 404" in FIG. 5. In addition, claim 1 further recites "creating a user status thread in a user entity of the target process." Creating the user status thread (UST) is illustrated, for example, by the arrow labeled "Create UST 406" in FIG. 6. For convenience of reference, FIGS. 5 and 6 are reproduced below.

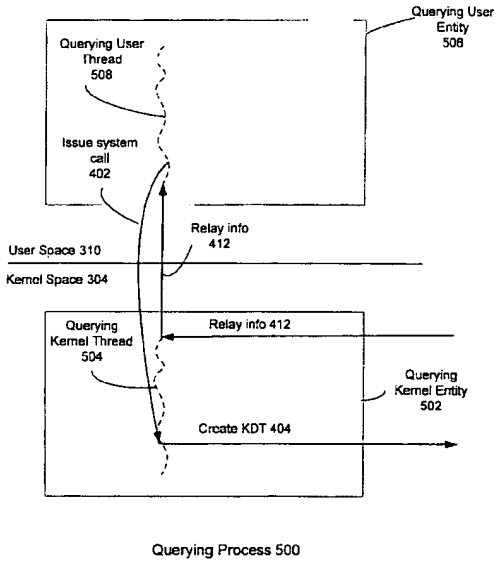

FIG. 5

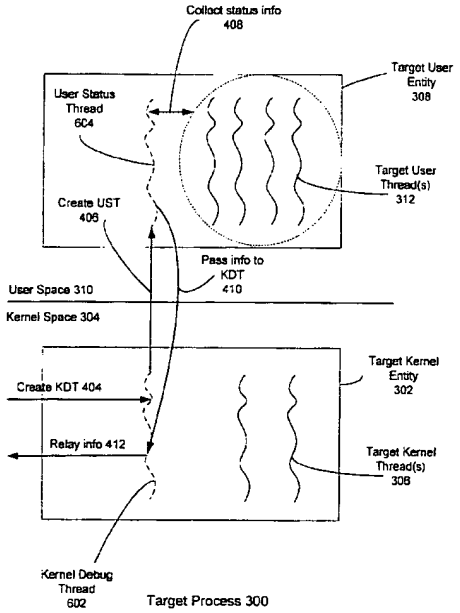

FIG. 6

As seen above, the step **404** of creating the KDT is initiated in kernel space **304** of the querying process **500** and results in a kernel debug thread **602** in the kernel space **304** of the target process **300**. The subsequent step **406** of creating the UST is initiated in kernel space **304** of the target process **300** and results in a user status thread **604** in the kernel space **304** of the target process **300**.

As discussed in the specification, a thread is composed of a context and a sequence of instructions to execute, where the context may comprise a register set and a program counter. (See page 1, lines 13-15.) As such, **creating a thread requires creating a context (including a register set and a program counter) and providing a sequence of instructions for the thread to execute.**

Applicant respectfully submits that McDowell does not disclose or suggest either the claimed step of creating a kernel debug thread, or the claimed step of creating a user status thread.

In relation to these claimed steps, citations are made in the office action to sections in McDowell relating to entry fields in the Multiprocessing Specification (MPS) table. As discussed on page 94 of McDowell, a portion of which is reproduced below for convenience of reference, the MPS table provides information about the logical layout of the hardware of a multiprocessing computer, such as the

routing between the various interrupts and the CPUs, and the addresses and resources consumed by the system buses.

The MPS Table

On a multiprocessing computer that's based on Intel processors, the operating system and BIOS must understand and agree on the logical layout of the hardware. The routing between the various interrupts and the CPUs must be defined. The addresses and resources consumed by the system buses must be understood. All of this information is defined in the MPS table.

MPS stands for *Multiprocessing Specification*. This is a specification put forth by Intel and Microsoft, among others, that defines the layout and content of this information. The latest revision of the MPS is version 1.4. It can be obtained from Intel's Developer web site (a treasure trove of hardware-related technical information) at *developer.intel.com*.

First, in regards to creating a KDT, a citation is made to the I/O APIC (input/output advanced programmable interrupt controller) Entry Field on page 97 of McDowell. The cited section is reproduced below for convenience of reference.

I/O APIC Entry Field

The next section details all of the I/O APICs in the system. In this case, there is only one:

```
io apic.   EN id 01 vers 11 @ fec00000
```

The first column indicates the entry type. For I/O APICs, the type is simply `io apic.`. The EN indicates that the APIC is enabled. At least one APIC in every system must be enabled.

Following this is the APIC ID. In this example, the APIC ID is 01. This APIC is version 11 (derived from bits 0–7 of the I/O APIC's version register). The base address of the APIC is 0xfec00000. This is the physical address of the I/O APIC.

However, as seen above, the I/O APIC Entry Field in the MPS table merely gives configuration information relating to the input/output APICs in the multiprocessor system. **Applicant respectfully submits that neither the existence nor the use of the I/O APIC Entry field discloses or suggests the claim limitation of "creating a kernel debug thread in a kernel entity of the target process." For example, the I/O APIC Entry Field is not usable to either create a context**

(including a register set and a program counter) or provide a sequence of instructions for the kernel debug thread to execute.

Second, in regards to creating a UST, a citation is made to the Processor Entry Fields on page 96 of McDowell. The cited section is reproduced below for convenience of reference.

> **Processor Entry Fields**
>
> For each processor in the system, there is a corresponding processor entry field describing the CPU. This entry is as follows:
>
> ```
> processor. EN BP L.APIC ID 00 Vers 11
>            Family 6, Model 3, Stepping 4, CPUID Flags 80fbff
> ```
>
> All entry fields begin with an identifying mnemonic. In this case, it's `processor`. The EN indicates that the processor is enabled. The system boot processor will follow the EN with a BP. There can be only one boot processor in a given system.
>
> The local APIC assigned to the CPU is detailed next. In this case, `L.APIC ID 00` means that local APIC 00 is the one assigned to the CPU. This is followed by the version of the local APIC—Version 11.
>
> The next line details the type and stepping information for the processor being described. Refer to Intel's web site for information relating this information back to a CPU type. What is important here is that all CPUs should have matching stepping numbers and CPUID flags. If they don't match, problems can occur when trying to run NT in multiprocessing mode.

However, as seen above, the Processor Entry Fields in the MPS table merely give configuration information relating to the CPUs in the multiprocessor system. **Applicant respectfully submits that neither the existence nor the use of the Processor Entry Fields in the MPS table disclose or suggest the claim limitation of "creating a user status thread in a user entity of the target process." For example, the Processor Entry Fields are not usable to either create a context (including a register set and a program counter) or provide a sequence of instructions for the user status thread to execute.**

For at least the above discussed reasons, claim 1 is patentably distinguished over the applied portions of the McDowell reference.

Claims 2-12 depend from claim 1. As such, for at least the same reasons as discussed above in relation to claim 1, claims 2-12 are also patentably distinguished over the applied portions of the McDowell reference.

Claims 13-15

Claim 13 is an independent claim which recites "a first system call configured to create a kernel debug thread in a kernel entity of the target process." Claim 13 further recites "a second system call configured to awake the kernel debug thread and pass information to the kernel debug thread."

First, in regards to creating a KDT, a citation is made to the Processor Entry Fields on page 96 of McDowell. The cited section is reproduced below for convenience of reference.

> **Processor Entry Fields**
>
> For each processor in the system, there is a corresponding processor entry field describing the CPU. This entry is as follows:
>
> ```
> processor. EN BP L.APIC ID 00 Vers 11
>            Family 6, Model 3, Stepping 4, CPUID Flags 80fbff
> ```
>
> All entry fields begin with an identifying mnemonic. In this case, it's `processor`. The EN indicates that the processor is enabled. The system boot processor will follow the EN with a BP. There can be only one boot processor in a given system.
>
> The local APIC assigned to the CPU is detailed next. In this case, `L.APIC ID 00` means that local APIC 00 is the one assigned to the CPU. This is followed by the version of the local APIC—Version 11.
>
> The next line details the type and stepping information for the processor being described. Refer to Intel's web site for information relating this information back to a CPU type. What is important here is that all CPUs should have matching stepping numbers and CPUID flags. If they don't match, problems can occur when trying to run NT in multiprocessing mode.

However, as seen above, the Processor Entry Fields in the MPS table merely give configuration information relating to the CPUs in the multiprocessor system. **Applicant respectfully submits that neither the existence nor the use of the Processor Entry Fields discloses or suggests the claim limitation of "a first system call configured to create a kernel debug thread in a kernel entity of the target process." For example, the Processor Entry Fields are not usable to either create a context (including a register set and a program counter) or provide a sequence of instructions for the kernel debug thread to execute.**

Second, in regards to creating a UST, a citation is made to the Interrupt Entry Fields on page 97 of McDowell. The cited section is reproduced below for convenience of reference.

Interrupt Entry Fields

After the I/O APIC entry field comes a series of lines describing interrupt routing across the APIC. These are called *I/O entry fields*. Let's use the following excerpt from the output for example:

```
io int.    intr   po=1 el=1, srcbus 02 irq 04 dst apic 01 intin 04
```

The first column, `io int.`, indicates that this entry describes an I/O interrupt. An I/O interrupt is an interrupt that is connected to the APIC interrupt input line. The other type of supported interrupt is a local interrupt. A local interrupt is one that is directly connected to a local interrupt input line on the APIC. The sample output above describes both types of interrupts.

The second column indicates the interrupt type described by this entry. The valid values for this field are `INTR` (a vectored interrupt, with the vector supplied by the APIC redirection table), `NMI` (nonmaskable interrupt), `SMI` (system management interrupt), and `ExtINT` (vectored interrupt from an external PIC).

The next column is labeled po. This describes the polarity of the underlying interrupt. The valid values are 0 for interrupts that conform to the underlying bus specification (e.g., active low for EISA, etc), 1 for active high, and 11 for active low.

However, as seen above, the Interrupt Entry Fields in the MPS table come after the I/O APIC Field and merely describe interrupt routing across the APIC. **Applicant respectfully submits that neither the existence nor the use of the Interrupt Entry Fields in the MPS table discloses or suggests the claim limitation of "a second system call configured to awake the kernel debug thread and pass information to the kernel debug thread."**

For at least the above discussed reasons, claim 13 is patentably distinguished over the applied portions of the McDowell reference.

Claims 14-15 depend from claim 13. As such, for at least the same reasons as discussed above in relation to claim 13, claims 14-15 are also patentably distinguished over the applied portions of the McDowell reference.

Conclusion

For the above-discussed reasons, applicant respectfully submits that claims 1-15 are now in form for allowance. Favorable action is respectfully requested.

The Examiner is also invited to call the below-referenced attorney to discuss this case.

Respectfully Submitted,

Weidong Cai

Dated: 11/29/2006

James K. Okamoto, Reg. No. 40,110
Okamoto & Benedicto LLP
P.O.Box 641330
San Jose, CA 95164-1330
Tel: (408) 436-2111
Fax: (408) 436-2114

CERTIFICATE OF MAILING			
I hereby certify that this correspondence, including the enclosures identified herein, is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. If the Express Mail Mailing Number is filled in below, then this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10.			
Signature:			
Typed or Printed Name:	James K. Okamoto	Dated:	11/29/2006
Express Mail Mailing Number (*optional*):			